

Um Pouco de Contentores em Java

João Paulo Barros

Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Beja, Portugal

Este texto apresenta, de forma muito resumida, os vários contentores, denominados *containers* em inglês e *collections* em Java, existentes no [Collections Framework](#) da linguagem Java.

Os contentores são agrupados numa tabela semelhante à apresentada [na documentação](#) da API da linguagem Java e acompanhados de uma breve descrição das suas características principais. Estas descrições, são de consulta rápida e pretendem ajudar na selecção do contentor mais adequado. No final encontra *links* para leituras adicionais sobre o tema.

		Acesso directo por chave	Array (acesso directo por índice) dinâmico	Elementos ordenados (os objectos inseridos têm de implementar a interface Comparable)	Acesso Sequencial + inserções no meio.	Acesso directo por chave + acesso sequencial
Interface	Set	HashSet		TreeSet		LinkedHashSet
	List		ArrayList		LinkedList	
	Map	HashMap		TreeMap		LinkedHashMap

- Implementações do tipo (interface) [Set](#) (permitem guardar conjuntos de elementos):
 - [HashSet](#). Impede duplicados. Pode encontrar um elemento rapidamente (bom para saber se ele lá está).
 - [TreeSet](#). Mantém os elementos ordenados e impede duplicados: pode ser visto como um conjunto ordenado. Podemos percorrer os elementos de forma ordenada mas não tem acesso directo por chave.
 - [LinkedHashSet](#). Parecido ao [HashSet](#) mas lembra-se da ordem pela qual os elementos foram inseridos. Também pode ser configurado para se lembrar da ordem pela qual os elementos foram acedidos.

- Implementações do tipo (interface) [List](#) (permitem guardar listas de elementos):
 - [ArrayList](#). Bom para inserir no fim (rápido) ou no meio (mais lentamente). Permite remover elementos e aceder a elementos numa dada posição. É a alternativa ao array.
 - [LinkedList](#). Melhor eficiência para inserir ou remover elementos no meio.
- Implementações do tipo (interface) [Map](#) (permitem guardar pares (key, value)):
 - [HashMap](#). Permite guardar pares (key, value) e aceder ao value, dado a key de forma directa e, tipicamente, muito rápida.
 - [TreeMap](#). Mantém os elementos ordenados e impede duplicados: pode ser visto como um conjunto ordenado. Podemos percorrer os elementos de forma ordenada e temos acesso a eles por chave, mas, tipicamente, de forma mais lenta do que o acesso directo do [HashMap](#) ou do [LinkedListHashMap](#).
 - [LinkedListHashMap](#). Parecido ao HashMap mas lembra-se da ordem pela qual os elementos foram inseridos. Também pode ser configurado para se lembrar da ordem pela qual os elementos foram acedidos.

Leituras complementares

“The Collections Framework”, disponível em

<http://java.sun.com/j2se/1.5.0/docs/guide/collections/index.html>

Joshua Bloch, “Trail: Collections”, disponível em

<http://java.sun.com/docs/books/tutorial/collections/index.html>

Tem comentários ou correcções a fazer a este texto? O que é que gostou mais? E menos? E do que é que não gostou mesmo nada? Os seus eventuais comentários ou correcções podem ser muito úteis para ajudar a melhorar futuras edições deste texto. Por si e também pelos seus futuros colegas, não se acanhe! Diga coisas para jpb@estig.ipbeja.pt.

Última revisão: Beja, 2 de Novembro de 2005

Autor: João Paulo Barros

Página pessoal: <http://www.estig.ipbeja.pt/~jpb>